Evaluation for the NeurIPS Machine Unlearning Competition

Written by the organizers of the NeurIPS Unlearning Competition*

August 2023

1 Introduction

In this document, we describe the evaluation procedure used in the NeurIPS Unlearning Competition. We begin by introducing a formal but non-worst case notion for unlearning that is largely inspired by [Sekhari et al., 2021, Gupta et al., 2021, Neel et al., 2021]. Next, we outline the procedure of defining an evaluation metric for forget quality based on that definition. Finally, we describe how we define our final score that, aside from forgetting quality, also takes into account model utility and efficiency.

Before we dive in, we would like to emphasize upfront that designing an evaluation metric for unlearning is as much a research project as the unlearning problem is itself. Our understanding of strong unlearning solutions evolves in tandem with our understanding of metrics to assess them. The best practices we adopt will very likely evolve over future iterations of the challenge as we learn from the competition's outcome. Therefore, we would like to emphasize that, while we hope our proposed evaluation framework is an important step forward for the community, we don't claim that it is perfect. The goal of this competition is to engage the community in discussions to understand how to improve both unlearning algorithms and evaluation metrics. See also the Limitations section at the end of this document for more details.

2 Background: Differential Privacy

Our definition of unlearning relates to the semantics of Differential Privacy (DP) . Therefore, we first recap key DP concepts, starting with the notion of *example-level differential privacy* (DP).

Definition 2.1. Example-level Differential Privacy. A training algorithm $A : \mathcal{D} \to \mathcal{R}$ is (ε, δ) example-level DP if for all pairs of datasets D and D' from $\mathcal{D}$ that differ by addition or removal of any single training example and all output regions $R \subseteq \mathcal{R}$:

$$\Pr[A(D) \in R] \leq e^{\varepsilon} \Pr[A(D') \in R] + \delta.$$

For example, if using a neural network with d parameters, the output space $\mathcal{R}$ is the d-dimensional Euclidean space in which the parameters live, and R can be any subset of it.

DP can be interpreted as a hypothesis test with the null hypothesis that A was trained on D and the alternative hypothesis that A was trained on D'. False positives (type-I errors) occur when the null hypothesis is true, but is rejected, while false negatives (type-II errors) occur when the alternative hypothesis is true, but is rejected. Kairouz et al. [2015] characterized (ε, δ)-DP in terms of the false positive rate (FPR) and false negative rate (FNR) achievable by an acceptance region. This characterization enables estimating the ε privacy parameter at a fixed δ as:

$$\hat{\varepsilon} = \max \left\{ \log \frac{1 - \delta - \hat{\text{FPR}}}{\hat{\text{FNR}}}, \log \frac{1 - \delta - \hat{\text{FNR}}}{\hat{\text{FPR}}} \right\}, \tag{1}$$

*If you have any questions, concerns, or feedback regarding this document, please direct them to the corresponding authors: Eleni Triantafillou (etriantafillou@google.com) and Peter Kairouz (kairouz@google.com).

where $\hat{\text{FPR}}$ and $\hat{\text{FNR}}$ are estimates of the true FPR and FNR under an instantiated *membership inference attack** that can inspect the model trained with DP in a black- or white-box fashion and attempts to infer whether the model was trained on D or D' [Shokri et al., 2017, Jagielski et al., 2020]. We observe that in order to obtain $\hat{\text{FPR}}$ and $\hat{\text{FNR}}$, the attack should be ran on many models trained with the same DP algorithm, some on D and others on D'. Moreover, if we would like to seek a high probability lower bound on ε, and we would like this lower bound to be as large as possible (i.e. as representative as possible), then we would need to do the following:

$$\hat{\varepsilon} = \sup_{\text{attacks}} \max \left\{ \log \frac{1-\delta-\overline{\text{FPR}}}{\underline{\text{FNR}}}, \log \frac{1-\delta-\overline{\text{FNR}}}{\underline{\text{FPR}}} \right\}, \tag{2}$$

where $\overline{\text{FPR}}$ ($\overline{\text{FNR}}$) and $\underline{\text{FNR}}$ ($\underline{\text{FPR}}$) are upper and lower bounds on FPR (FNR) and FNR (FPR), respectively, within a desired confidence level, and the sup is taken over a family of attacks. See [Steinke et al., 2023, Pillutla et al., 2023, Andrew et al., 2023, Nasr et al., 2023, 2021, Jagielski et al., 2020] for more details.

We now present a closely related notion, called group-level DP, which will be very useful when we present the unlearning definition in the next section.

Definition 2.2. Group-level Differential privacy. A training algorithm $A : \mathcal{D} \to \mathcal{R}$ is (ε, δ, k) group-level DP if for all pairs of datasets D and D' from $\mathcal{D}$ that differ by addition or removal of up to k training examples and all output regions $R \subseteq \mathcal{R}$:

$$\Pr[A(D) \in R] \leq e^{\varepsilon} \Pr[A(D') \in R] + \delta.$$

We note that the "group-level" DP notion differs from example-level DP in that it allows for the addition or removal of up to k arbitrary training examples, as opposed to only one. Thus, group-level DP can be stronger than example-level DP. Further, observe that group-level DP can also be interpreted as a binary hypothesis test, although the adversary can now incorporate the fact that D and D' differ by up to k training examples when making a decision.

3 Defining Machine Unlearning

We are now ready to define machine unlearning using the same semantics as DP. As mentioned above, our notion is largely inspired by [Sekhari et al., 2021, Gupta et al., 2021, Neel et al., 2021], although ours is a weaker version.

First, we define the notion of a *forget set* $S \subseteq D$. This is the set of training examples that we want the model trained on D to "forget". Intuitively, an *unlearning algorithm* $U(\cdot)$ ingests $A(D)$ (the model trained on D using algorithm A), the forget set S, and possibly the dataset D, and produces a model that has "forgotten" S. The notion of forgetting will be measured relative to running the training algorithm on $D \backslash S$ (i.e. $A(D \backslash S)$). This gives us the following mathematical definition, which draws inspiration from DP.

Definition 3.1. Machine Unlearning. For a fixed dataset D, forget set $S \subseteq D$, and a randomized learning algorithm A, an unlearning algorithm U is (ε, δ)-unlearning with respect to (D, S, A) if for all regions $R \subseteq \mathcal{R}$, we have that

$$\Pr[A(D \backslash S) \in R] \leq e^{\varepsilon} \Pr[U(A(D), S, D) \in R] + \delta,$$

and

$$\Pr[U(A(D), S, D) \in R] \leq e^{\varepsilon} \Pr[A(D \backslash S) \in R] + \delta.$$

Intuitively, when ε and δ are very small, the above definition says that the distributions of $A(D \backslash S)$ (the model retrained from scratch) and $U(A(D), S, D)$ (the unlearned model) are nearly indistinguishable from one another.

A few observations are in order.

*There is a long line of important research on membership inference attacks in ML, including many papers that apply these attacks to audit or estimate the ε of DP mechanisms. We apologize for not citing all the works in this space.

- Our definition is weaker than the one in [Sekhari et al., 2021, Gupta et al., 2021, Neel et al., 2021] as follows: (a) we do not consider adversarial (worst-case) datasets D, and (b) we do not consider adversarial (worst-case) forget sets S. In other words, our guarantee is restricted to a fixed dataset D and forget set S.
- Our definition of unlearning is strictly weaker than (ε, δ, k) group-level DP. In other words, if A satisfies (ε, δ, k) group-level DP, then for the *identity* unlearning function U, we automatically achieve (ε, δ)-unlearning with respect to any pair (D, S) as long as $|S| \leq k$. Thus, DP may be an overkill for accuracy if all we need is unlearning.
- As discussed in the previous section for DP, we can estimate the ε privacy parameter at a fixed δ using Equation (1); this time using estimates of FPR and FNR obtained for the appropriate hypothesis test for the unlearning case (i.e., the adversary can observe either the output of an unlearning algorithm or the output of an algorithm that retrains the model from scratch). For efficiency purposes, our competition will primarily use empirical estimates of FPR and FNR instead of high probability lower/upper bounds on these quantities (as done in Equation (2)). To ensure fairness in scoring, we might use high probability lower/upper bounds on FPR and FNR for the top submissions.

4 Defining and Measuring Forgetting Quality

We now discuss how we can use the above definition of unlearning to define an evaluation metric for unlearning algorithms.

Overview At a high level, we would like to measure how different two distributions are: the distribution obtained by the "oracle" unlearning algorithm of retraining from scratch ("retrain") without the forget set S, i.e. $A(D \backslash S)$, and the distribution obtained by a given unlearning algorithm U, i.e. $U(A(D), S, D)$, where A is a fixed training algorithm. We note that each of those is a distribution in the weight space, rather than a particular point, because randomness in the form of the initialization (in the case of retraining from scratch),

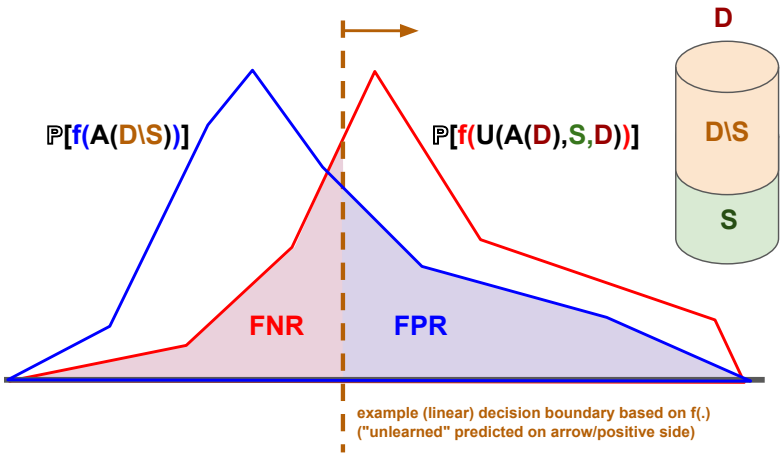


Figure 1: An illustration of FNR and FPR for a hypothesis test based on an example (linear) decision boundary and summary statistic, $f(\cdot)$. The plot depicts the densities of the summary statistic, computed on $A(D \setminus S)$ (i.e., on the output of the learning algorithm when training on the retain-set $D \setminus S$, alone), and on $U(A(D), S, D)$ (i.e., on the output of the unlearning procedure, $U(.)$, acting on the classifier learned on all data, $A(D)$, and the forget set, S). The unlearning level (ϵ, δ) achieved by the unlearning procedure can be estimated from the FNR and FPR of an optimal hypothesis test via Definition 3.1. Any choice for a hypothesis test can be interpreted as an attack, aiming to distinguish the learned and unlearned classifiers. Note that optimal hypothesis tests take the form of likelihood ratio tests, but any attack offers an estimate.

and the order in which data points appear in mini-batches (in both cases), may affect the resulting weights obtained by each procedure.

There are many different choices of attacks for separating two distributions. One could consider white-box attacks that operate on (estimates of) the two distributions in weight-space; though a difficulty inherent in that setup is the large dimensionality of the weight space. For ease of exposition in this document (and for the initial version for this competition), we consider black-box attacks that instead operate on (distributions of) *outputs* produced by retrained and unlearned models when receiving examples from the forget set S. For each example in the forget set, we run a number of attacks, each producing an estimate of FPR and FNR, and we keep the worst-case attack, from the perspective of the unlearning algorithm, i.e. the one that best separates the two distributions. We then use the estimated FPR and FNR of that worst-case attack to compute that example's ε estimate. We finally aggregate ε's across forget-set examples, to get an estimate of overall "forgetting quality".

Note that while one could in principle compute a single estimate of ε by comparing distributions of (unlearned and retrained) model outputs across forget set examples, we decided to compute one ε per example and then aggregate those ε values, as discussed below. The reason for this is that, as also discussed in [Carlini et al., 2022], different examples have different levels of "difficulty", causing the outputs of retrained and unlearned models to have different properties / take values in different ranges, which in turn makes it hard to reliably compare distributions of outputs of models that have been obtained by different examples.

Computing per-example ε's Let $\mathcal{R}$ and $\mathcal{U}$ denote the distributions of retrained and unlearned models, respectively (as noted above, these are distributions in weight space, obtained by training / unlearning with different random seeds). Further, for a particular forget set example $s \in S$, let $\mathcal{R}^s$ and $\mathcal{U}^s$ denote the distributions of (scalar) outputs of retrained and unlearned models, respectively, when receiving s as input.

We estimate $\mathcal{R}^s$ and $\mathcal{U}^s$ empirically by running retraining and unlearning N times each (with different random seeds) and compute two sets: $\mathcal{R}^s = \{f(R_1(s)), \dots f(R_N(s))\}$ and $\mathcal{U}^s = \{f(U_1(s)), \dots f(U_N(s))\}$, where R_i and U_i denote the i-th retrained and the i-th unlearned model, respectively, $M(s)$ yields the outputs obtained by feeding example s into model M, and f is a function that transforms those outputs into a scalar (the details of which we will reveal after the competition ends). Our evaluation metric is based on measuring how different the distributions of $\mathcal{R}^s$ and $\mathcal{U}^s$ are, for the different forget set examples s.

To that end, we define different "decision rules" ("attacks"), each of which makes a prediction about which of the two distributions ($\mathcal{R}^s$ or $\mathcal{U}^s$) a particular sample x (with $x \in \mathcal{R}^s \cup \mathcal{U}^s$) originates from. Each decision rule therefore yields a particular FPR and FNR (we will reveal the details of our decision rules after the competition ends). Intuitively, if a decision rule exists that can separate the two distributions well, then the given unlearning algorithm is not a good one, according to this metric. We therefore pick the decision rule that best separates the two distributions (i.e. the "worst-case" decision rule, from the perspective of the unlearning algorithm), and we use its FPR and FNR to compute the ε value for the particular forget example s, using Equation 1.

We show pseudocode for computing the ε value for a specific example in Algorithm 1. A few notes:

- If a particular attack fully separates the two distributions, we set the ε value for that attack to inf. We set this manually rather than using Equation 1 to avoid numerical issues.
- We chose to discard an attack if exactly one of its FPR and FNR is 0 (if both FPR and FNR are 0, we are in the "perfect separation" case covered above). This is because we hypothesize (based on empirical evidence) that such a situation is an artifact of the small number of samples we have from each distribution (we set $N = 512$); a choice that we are forced to make due to computational efficiency requirements.

Intuitively, ε^s, the ε value for a particular example s, measures the privacy degree, or the degree of indistinguishability between the distributions of the outputs of retrained and unlearned models for example s, where lower ε indicates better privacy / higher indistinguishability.

Aggregating per-example ε's Ultimately, we want a single estimate of an unlearning algorithm's "forgetting quality", which is an aggregation of the per-example ε's produced by that unlearning algorithm. To

Algorithm 1 This algorithm computes an example's ε from the FPRs and FNRs obtained by carrying out m attacks. Each of these attacks aims to distinguish the unlearned and retrained distributions of (transformed) outputs obtained from passing that example into N unlearned and N retrained models.

Require: FPR, FNR: two lists of length m each, storing the false positive and false negative rates (respectively) from running a collection of m attacks on the outputs of N unlearned and N retrained models, for a specific forget example.
Require: nan-max: a function that takes as input a list and returns the max of its elements, discarding any that are nan.
Require: δ: a float.
  $i \leftarrow 0$
  per-attack-$\varepsilon \leftarrow []$
  while $i \leq m$ **do**
    if FPR[i] $= 0$ and FNR[i] $= 0$ **then** ▷ This attack perfectly separates the two distributions.
      per-attack-$\varepsilon \leftarrow$ per-attack-$\varepsilon + [\text{inf}]$
    else if FPR[i] $= 0$ or FNR[i] $= 0$ **then** ▷ Discard attack if exactly one of FPR[i] or FNR[i] is 0.
      pass
    else ▷ Compute this attack's ε via Equation 1
      per-attack-$\varepsilon_1 \leftarrow \log(1 - \delta - \text{FPR[i]}) - \log(\text{FNR[i]}))$
      per-attack-$\varepsilon_2 \leftarrow \log(1 - \delta - \text{FNR[i]}) - \log(\text{FPR[i]}))$
      per-attack-$\varepsilon \leftarrow$ per-attack-$\varepsilon + [\text{nan-max}([\text{per-attack-}\varepsilon_1, \text{per-attack-}\varepsilon_2])\}]$
  $\varepsilon \leftarrow$ nan-max(per-attack-ε) ▷ The ε for this example is that of the strongest attack
  return ε

that end, we define a scoring function $\mathcal{H}$ that assigns a number of "points" to each example, based on that example's ε, and we aggregate by averaging the per-example scores. Specifically:

$$\mathcal{H}(s) = \frac{2}{2^{n(s)}}.$$

where n is a function that takes as input a forget example s and maps it to a "bin index" (an integer in the range $[1, B]$, where B is the total number of bins), based on its ε^s. Bins with smaller indices are better bins, to which better (smaller) ε^s values get assigned. We set the width of each bin to 0.5 and the total range of ε values we consider is $[0, 6.5)$, so $B = 13$. This is because we set $N = 512$, and with that N, we can't observe a ε beyond that range (where N is the number of models from each of the two distributions, retrained and unlearned, as discussed above).

So, for example, $n(s)$ is 1 if ε^s is in the first bin, i.e. $\varepsilon^s \in [0, 0.5)$, $n(s)$ is 2 if ε^s is in the second bin, i.e. $\varepsilon^s \in [0.5, 1)$, and so on. For illustration, see below the associated $\mathcal{H}$ value that a forget example s would receive depending on which bin its ε^s falls in (showing only the first few bins for illustration).

$$\mathcal{H}(s) = \begin{cases} 1 & 0.0 \leq \varepsilon^s < 0.5 \\ 0.5 & 0.5 \leq \varepsilon^s < 1.0 \\ 0.25 & 1.0 \leq \varepsilon^s < 1.5 \\ 0.125 & 1.5 \leq \varepsilon^s < 2.0 \\ \dots & \end{cases}$$

Note that $\mathcal{H}(s)$ assigns higher points for lower ε (recall that the smaller the ε for an example, the more indistinguishable the retrained and unlearned distributions are for that example, which is indicative of successful unlearning). †

†We considered several other ways of aggregating across per-example ε values. An alternative idea was returning the maximum ε across forget set examples, but we worried that that would give too pessimistic an estimate and won't be able to distinguish well between different unlearning algorithms (while many may be similar in the worst-case, their distributions of ε values over examples might be different). Computing quantiles over ε values would also be possible, but we decided that the current proposal is a more granular way of comparing unlearing algorithms to one another. We hope that future iterations of our evaluation protocol will improve upon this choice.

Finally, we define the forgetting quality for a given unlearning algorithm as the average score over the examples of the forget set:

$$\mathcal{F} = \frac{1}{|S|} \sum_{s \in S} \mathcal{H}(s).$$

5 Overall Scoring: Forgetting Quality, Efficiency and Utility

A good unlearning algorithm must naturally ensure high forget quality while being frugal in terms of resource usage (efficiency) and not damaging model utility. Hence, in this section, we discuss how we combine forgetting quality, as defined in the previous section, with 1) model utility and 2) efficiency.

Computing model utility We compute the utility of an unlearning algorithm via the accuracy of unlearned models on the "retain set" $D \backslash S$ and a held-out test set T. Concretely, we compute the average Retain Accuracy (RA) and Test Accuracy (TA) as follows, where the average is taken over a set O of models:

$$RA(O) = \frac{1}{|O|} \sum_{i \in 1,\ldots,|O|} \text{Acc}(O_i, D \backslash S), \qquad TA(O) = \frac{1}{|O|} \sum_{i \in 1,\ldots,|O|} \text{Acc}(O_i, T),$$

where $\text{Acc}(M, D)$ denotes the accuracy of model M on dataset D.

For convenience, let us define:

$$RA^U = RA(\{U_1, \ldots U_N\}).$$
$$TA^U = TA(\{U_1, \ldots U_N\}).$$
$$RA^R = RA(\{R_1, \ldots R_N\}).$$
$$TA^R = TA(\{R_1, \ldots R_N\}).$$

where RA^U and TA^U are the estimates of the retain and test accuracy, respectively, of unlearned models and similarly, RA^R and TA^R are the estimates of the retain and test accuracy, respectively, for the ideal unlearning algorithm of retraining. We consider that an unlearning algorithm has good utility when RA^U and TA^U are close to their RA^R and TA^R counterparts.

Combining forget quality, utility and efficiency First, to take efficiency into account, we reject unlearning algorithms that run slower than a pre-decided hard threshold (in terms of seconds). This hard threshold is chosen to be a small percentage (about 20%) of the time that it takes to run retrain-from-scratch. The rationale behind this choice is that, to justify incurring the cost of unlearning (which will unavoidably have suboptimal forget quality in general), running unlearning must be significantly faster than retraining.

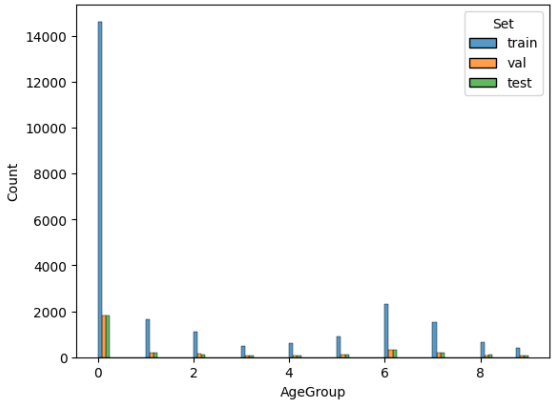
Then, for unlearning algorithms that pass the efficiency cut-off, we compute their final score as follows (where higher is better):

$$\mathcal{F} \times \frac{RA^U}{RA^R} \times \frac{TA^U}{TA^R}.$$

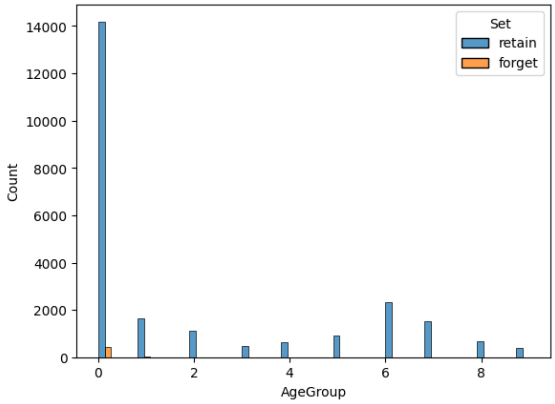
Intuitively, the above formula adjusts the forgetting quality $\mathcal{F}$ based on utility, by penalizing an unlearning algorithm if either its retain or test (average) accuracy is smaller than the corresponding average accuracy of retraining. This score rewards unlearning algorithms that yield both high forgetting quality as well as good utility.

6 Experimental Setup

Dataset details and forget set split We use a dataset of natural images of people's faces. Each image is labelled with an age group (there are 10 total classes / age groups). We split the dataset into a training, validation and test set. We further split the training set into a retain set and a forget set. When doing so, we take care that no subject's images are split between the retain and the forget set; that is, each subject is placed entirely in either the retain set or the forget set. The size of the forget set is roughly 2% of the size of the training set. In Figure 2, we show the class (age group) distribution of the different sets.



(a) Train, validation and test histograms



(b) Retain and forget histograms

Figure 2: Histograms of age groups for different sets. As can be seen, the distribution across classes / age groups is similar across the train, validation and test sets. There is a lot of class imbalance: class 0 is by far the most common. Note too that, while the retain set follows a similar distribution as the training set, the forget set contains examples from only the first two classes, with the vast majority belonging to class 0.

Training details The "original model" we consider is a ResNet-18 classifier, trained for 30 epochs on the training set to predict the age group associated with each image of a person's face. It is trained with class weights, to deal with class imbalance (where the loss value of an example is adjusted based on how frequent that example's class label is). We use no data augmentation. The original model obtains 98.98% accuracy on the training set and 96.43% on the test set.

Public and private leaderboards The results displayed on the public leaderboard will be obtained using the train / valid / test / retain / forget splits (and corresponding original model) mentioned above, whose class distribution is shown in Figure 2. However, we will have a second split (following the same properties as the one described above, but a different random seed) that we will use to declare final winners, using a private leaderboard. This is common practice in competitions and the rationale is to prevent "overfitting" to the particular setup for which feedback is provided on the leaderboard.

7 Disclaimer and Limitations

As mentioned, the key driving motivation for this competition is to engage the community in investigations and discussions that will shed light and push the envelope in this very important and nascent area. Pragmatically and by design, this entails deepening our understanding on how to design and implement better unlearning algorithms, as well as how to design appropriate metrics for their evaluation. The evaluation protocol of the challenge has been implemented with a high level of care, combining expertise from key areas such as deep learning, unlearning algorithms, differential privacy, etc. Our proposed evaluation metric aims to (i) align key concepts from DP with machine unlearning evaluation for forget quality, (ii) strike a balance between the ability of the metric for evaluating forget quality and its utility and efficiency, and (iii) for practical reasons (owing to this being a Kaggle-hosted competition), combine forget quality, model utility, and unlearning algorithm efficiency into a single score. As a result, no guarantees can be provided by the organizers that our evaluation metric is flawless, nor that the results of the challenge will be indicative of the fitness of the algorithms to perform unlearning in other settings.

Specifically, assessing and quantifying forgetting quality of an unlearning algorithm is unavoidably fraught with formidable challenges: formally defining measures for forgetting quality, as well as reliably estimating those in practice are both open research problems. Notably, reliably measuring indistinguishability between distributions is very challenging to do in a computationally-efficient manner. We believe that these are important areas for future work and hope that our proposed approach inspires further research in this

area. To gain a clearer understanding of the properties of our proposed metric, in our post-competition analyses we may investigate how it correlates with different metrics used in the literature for forgetting quality. Admittedly, we chose to use a relatively smaller-than-desired number of samples from each distribution. This was done in order to avoid impractically-long running times required to score each submission to the competition. However, to mitigate the noise and limitations of the above, we may also choose to run the top 5 submissions with a substantially larger number of models, after the competition ends.

Further, our procedure for combining forgetting quality with utility and efficiency is also unavoidably lossy (as this tends to lose key information about the performance of an algorithm in a specific area of interest (e.g., in test or retain accuracy, or in forget quality): it is not possible to produce a single score that fully captures all of this information (note that this is common in evaluating problems with multiple objectives / desiderata and isn't unique to our competition). To mitigate this, in our post-competition analyses and subsequent iterations of this competition, we may report separately on the performance of algorithms across these combined sub-metrics and may even design alternative ways to better understand how well different algorithms deal with trade-offs like that between forgetting quality and efficiency, and that between forgetting quality and utility.

Finally, unlearning is a young area of research and there are many aspects of the problem that are left for future exploration. For example, an interesting scenario that our setup disallows is the ability to modify the original model training in order to produce trained models that are more amenable to unlearning. Other interesting investigations involve understanding the ability of different algorithms to unlearn different forget sets (both in terms of size and composition), to investigate the effect of different pre-training algorithms, architectures, class imbalance and data augmentation, to name a few. Our competition unavoidably limits the scope to studying a particular scenario (dataset / architecture / forget set, etc). We also limit to considering only one value for δ, whereas one could consider a trade-off curve for ε and δ values. To mitigate these limitations, in our post-competition analyses we may run top-performing unlearning algorithms in different settings and consider different scenarios in follow-up iterations of the unlearning competition.

8 Acknowledgements

We thank Jamie Hayes and Sewoong Oh for insightful discussions and for their valuable feedback and suggestions on this document.

References

G. Andrew, P. Kairouz, S. Oh, A. Oprea, H. B. McMahan, and V. Suriyakumar. One-shot empirical privacy estimation for federated learning. *arXiv preprint arXiv:2302.03098*, 2023.

N. Carlini, S. Chien, M. Nasr, S. Song, A. Terzis, and F. Tramer. Membership inference attacks from first principles. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 1897–1914. IEEE, 2022.

V. Gupta, C. Jung, S. Neel, A. Roth, S. Sharifi-Malvajerdi, and C. Waites. Adaptive machine unlearning. *Advances in Neural Information Processing Systems*, 34:16319–16330, 2021.

M. Jagielski, J. Ullman, and A. Oprea. Auditing differentially private machine learning: How private is private sgd? *Advances in Neural Information Processing Systems*, 33:22205–22216, 2020.

P. Kairouz, S. Oh, and P. Viswanath. The composition theorem for differential privacy. In *International conference on machine learning*, pages 1376–1385. PMLR, 2015.

M. Nasr, S. Songi, A. Thakurta, N. Papernot, and N. Carlin. Adversary instantiation: Lower bounds for differentially private machine learning. In *2021 IEEE Symposium on security and privacy (SP)*, pages 866–882. IEEE, 2021.

M. Nasr, J. Hayes, T. Steinke, B. Balle, F. Tramèr, M. Jagielski, N. Carlini, and A. Terzis. Tight auditing of differentially private machine learning. *arXiv preprint arXiv:2302.07956*, 2023.

S. Neel, A. Roth, and S. Sharifi-Malvajerdi. Descent-to-delete: Gradient-based methods for machine unlearning. In *Algorithmic Learning Theory*, pages 931–962. PMLR, 2021.

K. Pillutla, G. Andrew, P. Kairouz, H. B. McMahan, A. Oprea, and S. Oh. Unleashing the power of randomization in auditing differentially private ml. *arXiv preprint arXiv:2305.18447*, 2023.

A. Sekhari, J. Acharya, G. Kamath, and A. T. Suresh. Remember what you want to forget: Algorithms for machine unlearning. *Advances in Neural Information Processing Systems*, 34:18075–18086, 2021.

R. Shokri, M. Stronati, C. Song, and V. Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*, pages 3–18. IEEE, 2017.

T. Steinke, M. Nasr, and M. Jagielski. Privacy auditing with one (1) training run. *arXiv preprint arXiv:2305.08846*, 2023.